

P&S Mobile Genomics

Lecture 12: GenASM

Dr. Damla Senol Cali

ETH Zurich

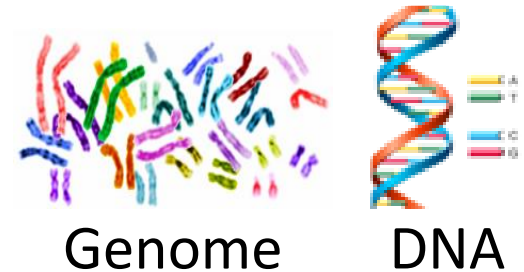
Fall 2022

16 January 2023

Genome Sequencing

- **Genome sequencing:** Enables us to determine the order of the DNA sequence in an organism's genome
 - Plays a **pivotal role** in:
 - Personalized medicine
 - Outbreak tracing
 - Understanding of evolution

- Modern genome sequencing machines extract smaller randomized fragments of the original DNA sequence, known as **reads**
 - *Short reads:* a few hundred base pairs, error rate of ~0.1%
 - *Long reads:* thousands to millions of base pairs, error rate of 10–15%



Genome Sequence Analysis

- ❑ **Read mapping:** *First key step* in genome sequence analysis (GSA)
 - Aligns **reads** to one or more possible locations within the **reference genome**, and
 - Finds the **matches** and **differences** between the read and the reference genome segment at that location

- ❑ Multiple steps of read mapping require ***approximate string matching***
 - Approximate string matching (ASM) enables read mapping to account for **sequencing errors** and **genetic variations** in the reads

- ❑ Bottlenecked by the **computational power and memory bandwidth limitations of existing systems**

GenASM: ASM Framework for GSA

Our Goal:

Accelerate approximate string matching
by designing a fast and flexible framework,
which can accelerate *multiple steps* of genome sequence analysis

- ❑ **GenASM:** First ASM acceleration framework for GSA
 - Based upon the *Bitap* algorithm
 - Uses fast and simple bitwise operations to perform ASM
 - Modified and extended ASM algorithm
 - Highly-parallel Bitap with long read support
 - Novel bitvector-based algorithm to perform *traceback*
 - Co-design of our modified scalable and memory-efficient algorithms with low-power and area-efficient hardware accelerators

Use Cases & Key Results

(1) Read Alignment

- ❑ **116×** speedup, **37×** less power than **Minimap2** (state-of-the-art **SW**)
- ❑ **111×** speedup, **33×** less power than **BWA-MEM** (state-of-the-art **SW**)
- ❑ **3.9×** better throughput, **2.7×** less power than **Darwin** (state-of-the-art **HW**)
- ❑ **1.9×** better throughput, **82%** less logic power than **GenAx** (state-of-the-art **HW**)

(2) Pre-Alignment Filtering

- ❑ **3.7×** speedup, **1.7×** less power than **Shouji** (state-of-the-art **HW**)

(3) Edit Distance Calculation

- ❑ **22–12501×** speedup, **548–582×** less power than **Edlib** (state-of-the-art **SW**)
- ❑ **9.3–400×** speedup, **67×** less power than **ASAP** (state-of-the-art **HW**)

Outline

❑ Introduction

❑ Background

- **Genome Sequencing & Genome Sequence Analysis**
- **Approximate String Matching (ASM)**
- **ASM with Bitap Algorithm**

❑ GenASM: ASM Acceleration Framework

- GenASM Algorithm
- GenASM Hardware Design
- Use Cases of GenASM

❑ Evaluation

❑ Conclusion

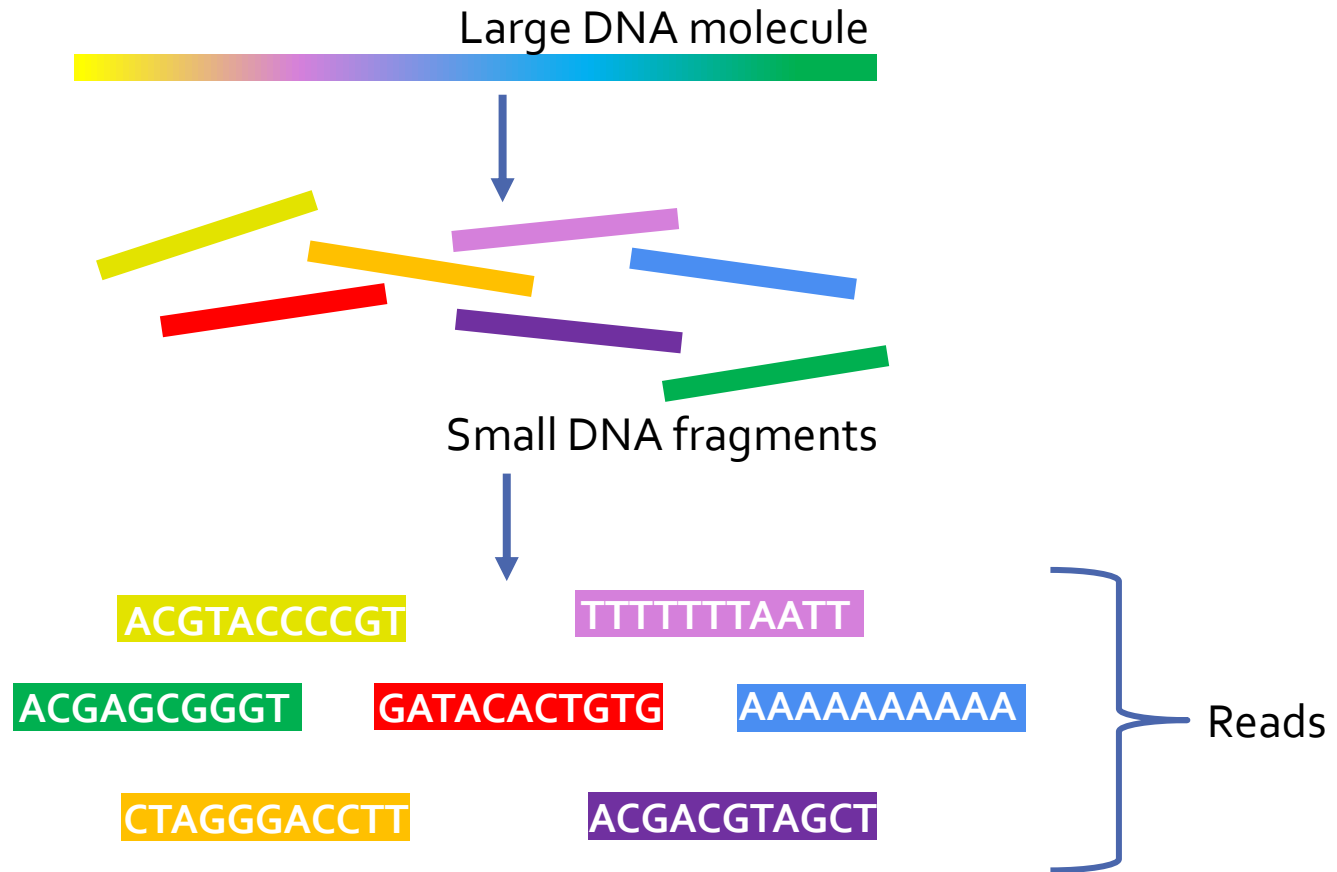
Genome Sequencing

- ❑ Goal is to determine the order of the DNA sequence (composed of A, C, G, Ts) in an organism's genome

- ❑ **Challenges:**

- There is no machine that takes long DNA as an input, and gives the complete sequence as output
- All sequencing machines chop DNA into pieces and identify relatively small pieces (but not how they fit together)

Genome Sequencing (cont'd.)



Sequencing of COVID-19

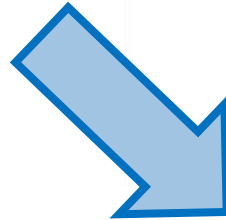
Why genome sequencing and sequence data analysis are important?

- ❑ To detect the virus from a human sample
- ❑ To understand the sources and modes of transmission of the virus
- ❑ To sequence the genome of the virus itself, COVID-19, in order to track the mutations in the virus
- ❑ To explore the genes of infected patients
 - To understand why some people get more severe symptoms than others
 - To help with the development of new treatments

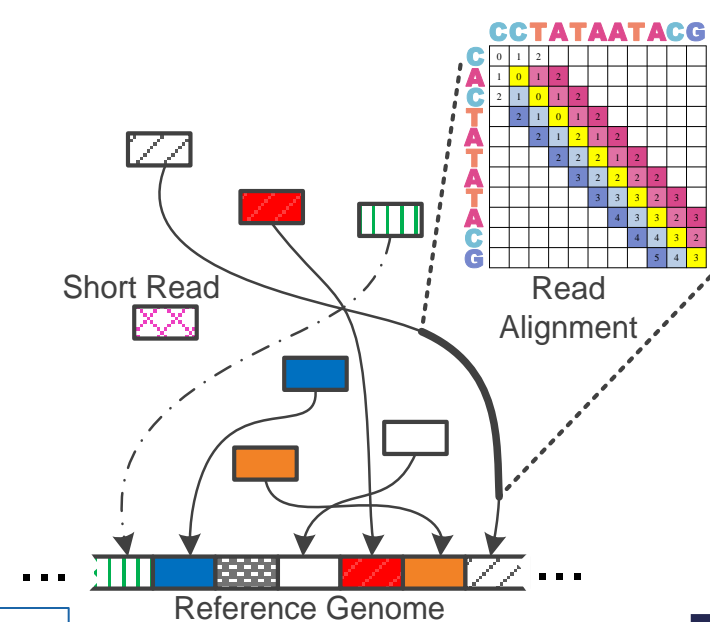
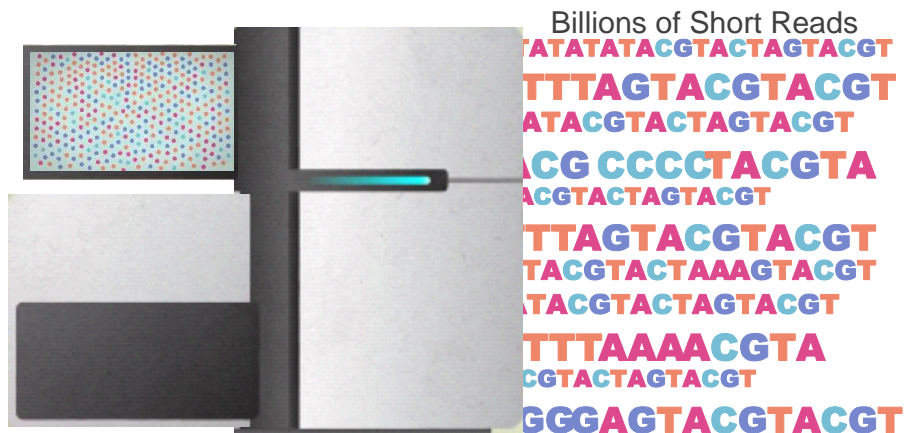
Future of Genome Sequencing & Analysis



MinION from ONT



SmidgION from ONT



1 Sequencing

Genome Analysis

2 Read Mapping

reference: TTTATCGCTTCCATGACGCAG

read1: ATCGCATCC

read2: TATCGCATC

read3: CATCCATGA

read4: CGCTTCCAT

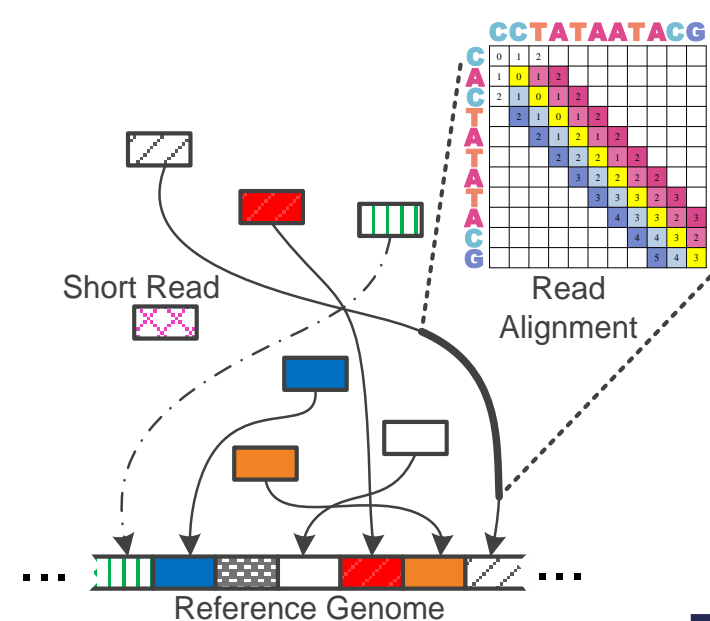
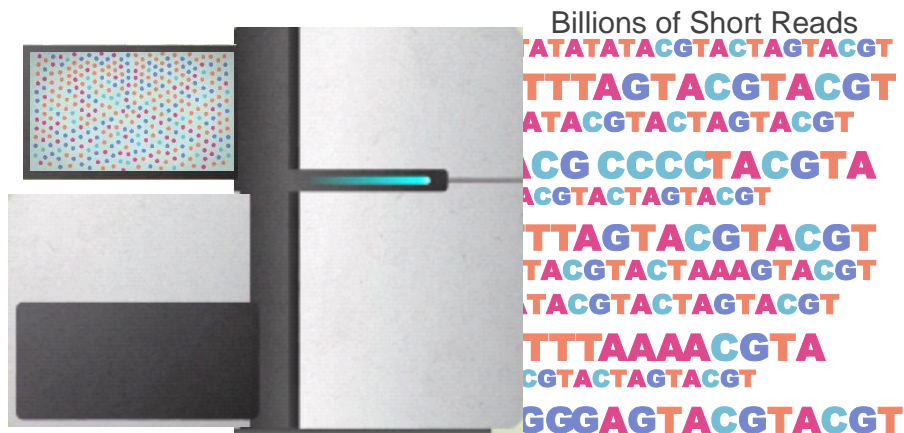
read5: CCATGACGC

read6: TTCCATGAC

3 Variant Calling



4 Scientific Discovery



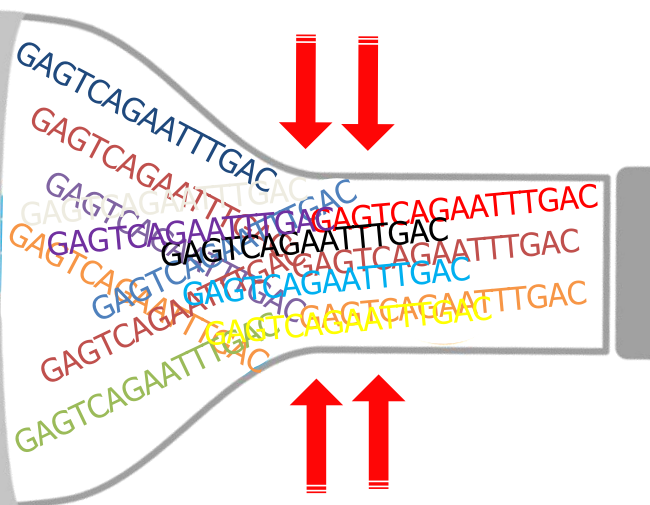
1 Sequencing

Read Mapping 2

Bottlenecked in Mapping!!

Illumina HiSeq4000

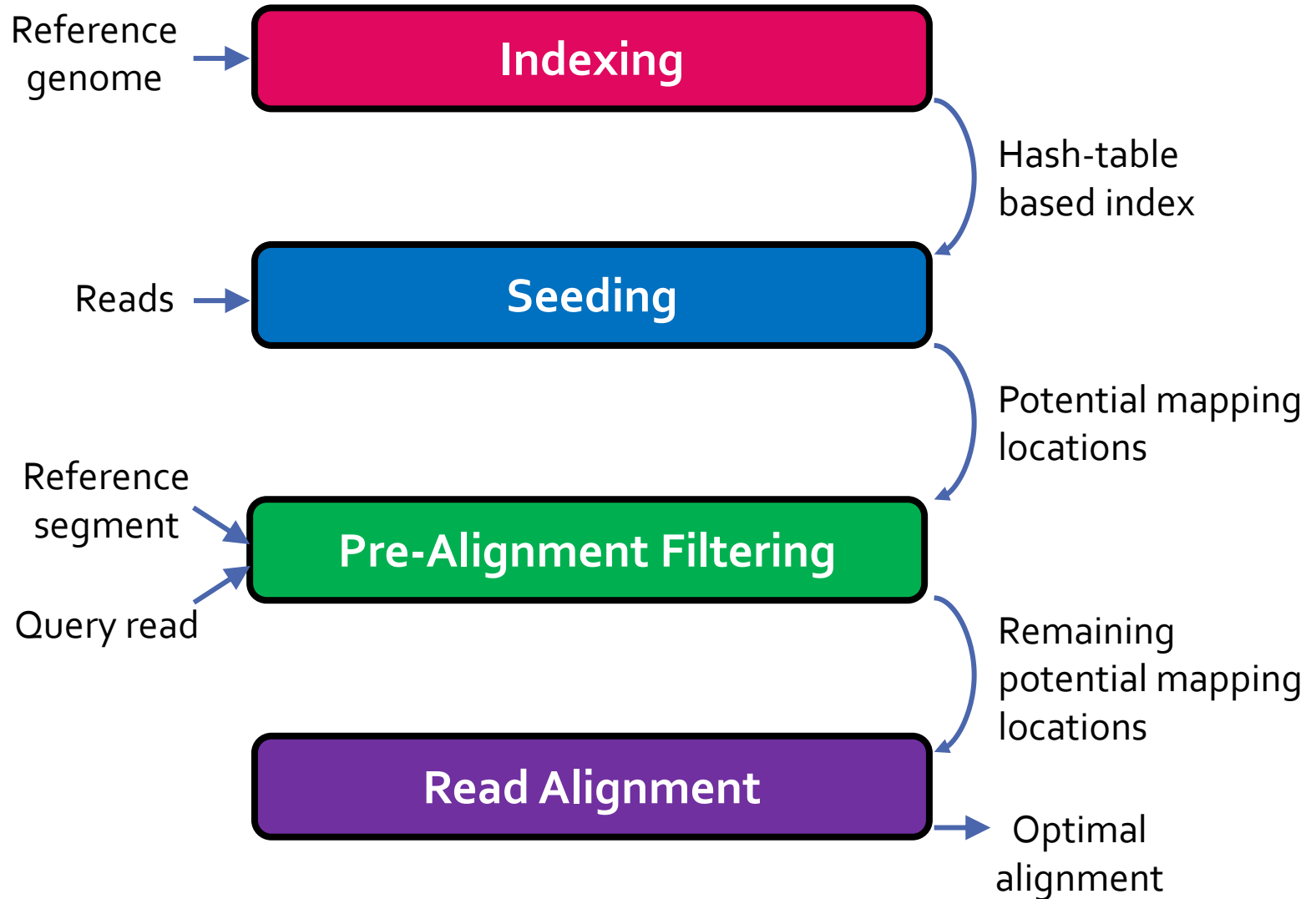
300 M
bases/min



on average

2 M
bases/min
(0.6%)

Read Mapping



Approximate String Matching

- Sequenced genome **may not exactly map** to the reference genome due to **genetic variations** and **sequencing errors**

Reference: AAAA**A**TGTTTA**G**TGCTAC**T**TG
Read: AAA**T**GTTTA**C**TGCTAC**T**TG
deletion *substitution* *insertion*

- Approximate string matching (ASM):**
 - Detect the **differences** and **similarities** between two sequences
 - In genomics, ASM is required to:
 - Find the *minimum edit distance* (i.e., total number of edits)
 - Find the *optimal alignment* with a *traceback* step
 - Sequence of matches, substitutions, insertions and deletions, along with their positions
 - 3M-1D-6M-1S-6M-1I-2M** for the above example
 - Usually implemented as a **dynamic programming (DP) based algorithm**

Bitap Algorithm

- ❑ Bitap^{1,2} performs ASM with fast and simple bitwise operations
 - Amenable to efficient hardware acceleration
 - Computes the **minimum edit distance** between a **text** (e.g., reference genome) and a **pattern** (e.g., read) with a maximum of k errors
- ❑ **Step 1: Pre-processing (per pattern)**
 - Generate a **pattern bitmask (PM)** for each character in the alphabet (A, C, G, T)
 - Each PM indicates if character exists at each position of the pattern
- ❑ **Step 2: Searching (Edit Distance Calculation)**
 - Compare all characters of the text with the pattern by using:
 - Pattern bitmasks
 - Status bitvectors that hold the partial matches
 - Bitwise operations

[1] R. A. Baeza-Yates and G. H. Gonnet. "A New Approach to Text Searching." *CACM*, 1992.

[2] S. Wu and U. Manber. "Fast Text Searching: Allowing Errors." *CACM*, 1992.

Bitap Algorithm (cont'd.)

□ Step 2: Edit Distance Calculation

For each character of the text (char):

Copy previous R bitvectors as oldR

$R[0] = (\text{oldR}[0] \ll 1) \mid \text{PM}[\text{char}]$

For $d = 1 \dots k$:

deletion = $\text{oldR}[d-1]$

substitution = $\text{oldR}[d-1] \ll 1$

insertion = $R[d-1] \ll 1$

match = $(\text{oldR}[d] \ll 1) \mid \text{PM}[\text{char}]$

$R[d] = \text{deletion} \& \text{mismatch} \& \text{insertion} \& \text{match}$

Check MSB of $R[d]$:

If 1, no match.

If 0, match with d many errors.

Bitap Algorithm (cont'd.)

□ Step 2: Edit Distance Calculation

For each character of the text (char):

Copy previous R bitvectors as oldR

$R[0] = (\text{oldR}[0] \ll 1) \mid \text{PM}[\text{char}]$

For $d = 1 \dots k$:

deletion = $\text{oldR}[d-1]$

substitution = $\text{oldR}[d-1] \ll 1$

insertion = $R[d-1] \ll 1$

match = $(\text{oldR}[d] \ll 1) \mid \text{PM}[\text{char}]$

$R[d] = \text{deletion} \& \text{mismatch} \& \text{insertion} \& \text{match}$

Check MSB of $R[d]$:

If 1, no match.

If 0, match with d many errors.

Large number of iterations

Bitap Algorithm (cont'd.)

□ Step 2: Edit Distance Calculation

For each character of the text (char):

Copy previous R bitvectors as oldR

$R[0] = (\text{oldR}[0] \ll 1) \mid \text{PM}[\text{char}]$

For $d = 1 \dots k$:

deletion	$= \text{oldR}[d-1]$
substitution	$= \text{oldR}[d-1] \ll 1$
insertion	$= R[d-1] \ll 1$
match	$= (\text{oldR}[d] \ll 1) \mid \text{PM}[\text{char}]$

$R[d] = \text{deletion} \ \& \ \text{mismatch} \ \& \ \text{insertion} \ \& \ \text{match}$

Check MSB of $R[d]$:

If 1, no match.

If 0, match with d many errors.

Data dependency
between iterations
(i.e., no
parallelization)

Bitap Algorithm (cont'd.)

□ Step 2: Edit Distance Calculation

For each character of the text (char):

Copy previous R bitvectors as oldR

$R[0] = (\text{oldR}[0] \ll 1) \mid \text{PM}[\text{char}]$

For $d = 1 \dots k$:

deletion = $\text{oldR}[d-1]$

substitution = $\text{oldR}[d-1] \ll 1$

insertion = $R[d-1] \ll 1$

match = $(\text{oldR}[d] \ll 1) \mid \text{PM}[\text{char}]$

$R[d] = \text{deletion} \ \& \ \text{mismatch} \ \& \ \text{insertion} \ \& \ \text{match}$

Check MSB of $R[d]$:

If 1, no match.

If 0, match with d many errors.

Does *not* store and process these intermediate bitvectors to find the optimal alignment (i.e., no traceback)

Limitations of Bitap

1) Data Dependency Between Iterations:

Algorithm

- Two-level data dependency forces the consecutive iterations to take place sequentially

2) No Support for Traceback:

- Bitap does not include any support for optimal alignment identification

3) No Support for Long Reads:

- Each bitvector has a length equal to the length of the pattern
- Bitwise operations are performed on these bitvectors

4) Limited Compute Parallelism:

Hardware

- Text-level parallelism
- Limited by the number of compute units in existing systems

5) Limited Memory Bandwidth:

- High memory bandwidth required to read and write the computed bitvectors to memory

Outline

- ❑ Introduction
- ❑ Background
 - Genome Sequencing & Genome Sequence Analysis
 - Approximate String Matching (ASM)
 - ASM with Bitap Algorithm
- ❑ **GenASM: ASM Acceleration Framework**
 - **GenASM Algorithm**
 - **GenASM Hardware Design**
 - **Use Cases of GenASM**
- ❑ Evaluation
- ❑ Conclusion

GenASM: ASM Framework for GSA

- ❑ Approximate string matching (ASM) acceleration framework based on the Bitap algorithm
- ❑ *First ASM acceleration framework* for genome sequence analysis
- ❑ We overcome the *five limitations* that hinder Bitap's use in genome sequence analysis:
 - Modified and extended ASM algorithm
 - Highly-parallel Bitap with long read support
 - Novel bitvector-based algorithm to perform *traceback*
 - Specialized, low-power and area-efficient hardware for both modified Bitap and novel traceback algorithms

GenASM Algorithm

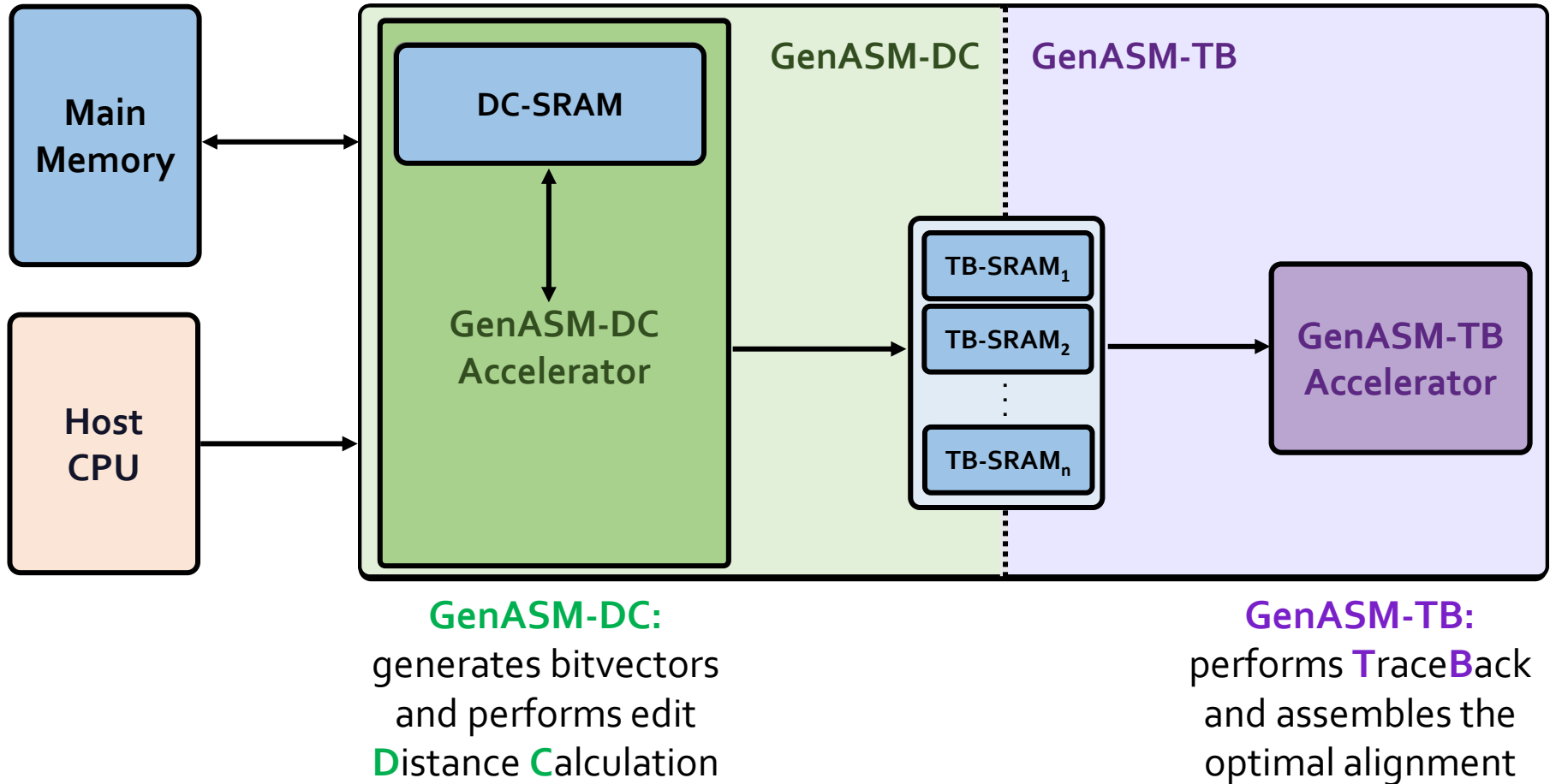
❑ GenASM-DC Algorithm:

- Modified Bitap for Distance Calculation
- Extended for efficient long read support
- Besides bit-parallelism that Bitap has, extended for parallelism:
 - Loop unrolling
 - Text-level parallelism

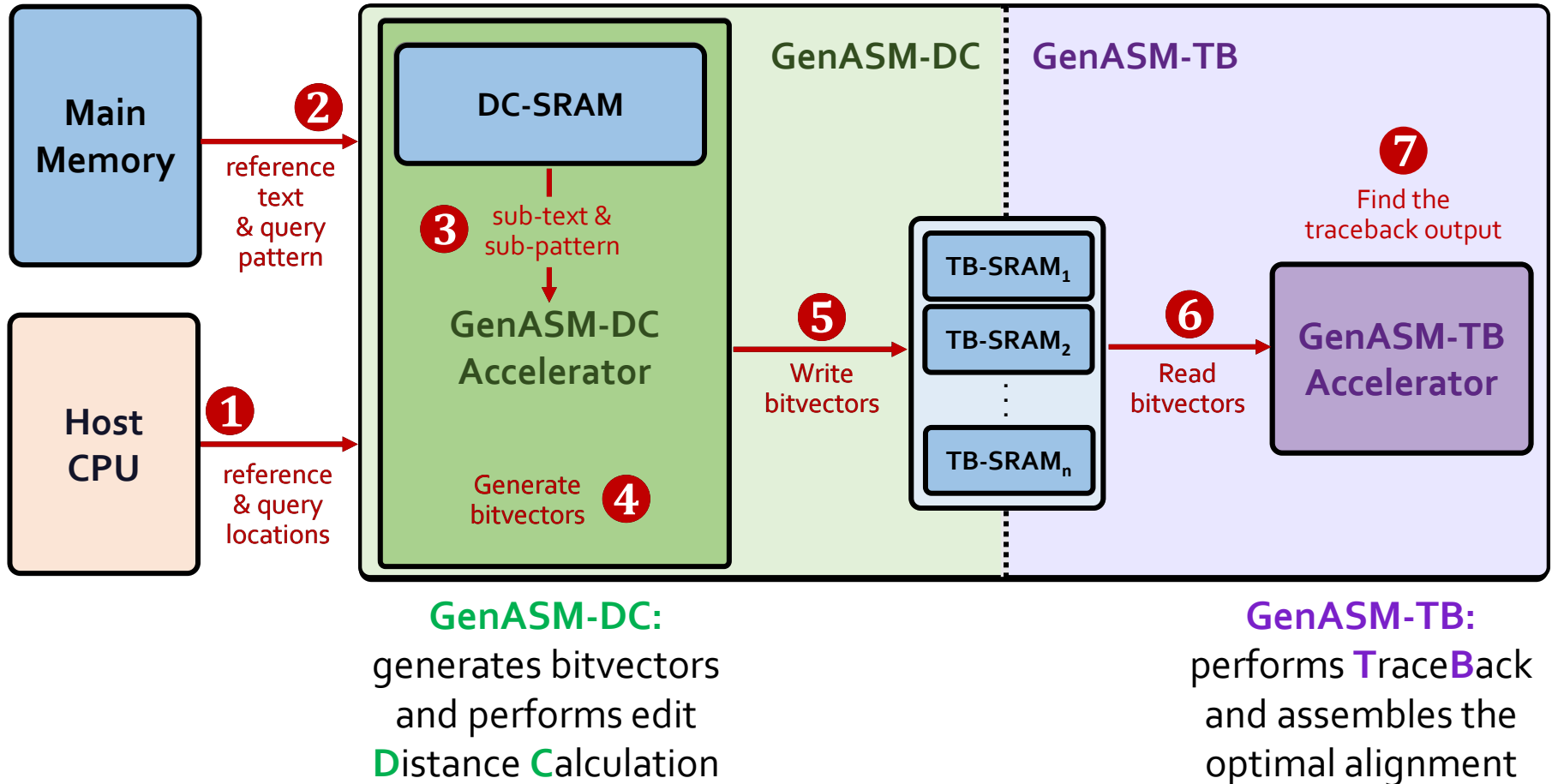
❑ GenASM-TB Algorithm:

- Novel Bitap-compatible TraceBack algorithm
- Walks through the intermediate bitvectors (match, deletion, substitution, insertion) generated by GenASM-DC
- Follows a divide-and-conquer approach to decrease the memory footprint

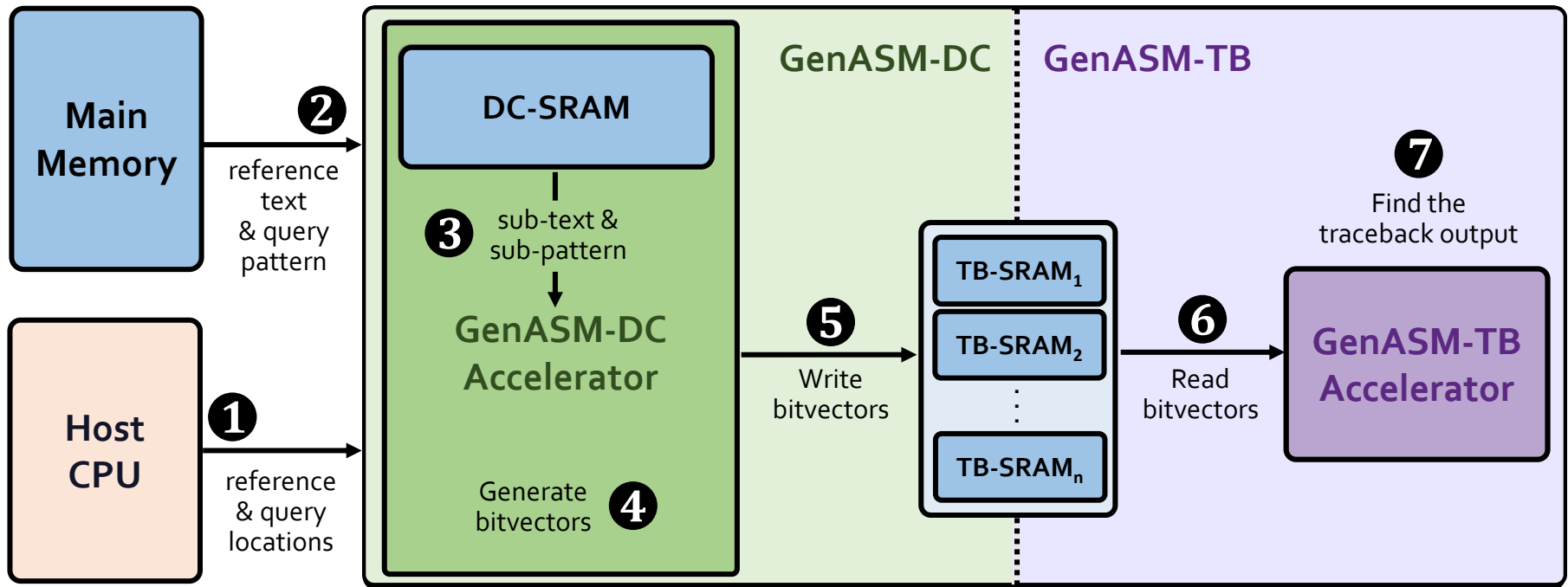
GenASM Hardware Design



GenASM Hardware Design



GenASM Hardware Design



Our *specialized compute units* and *on-chip SRAMs* help us to:

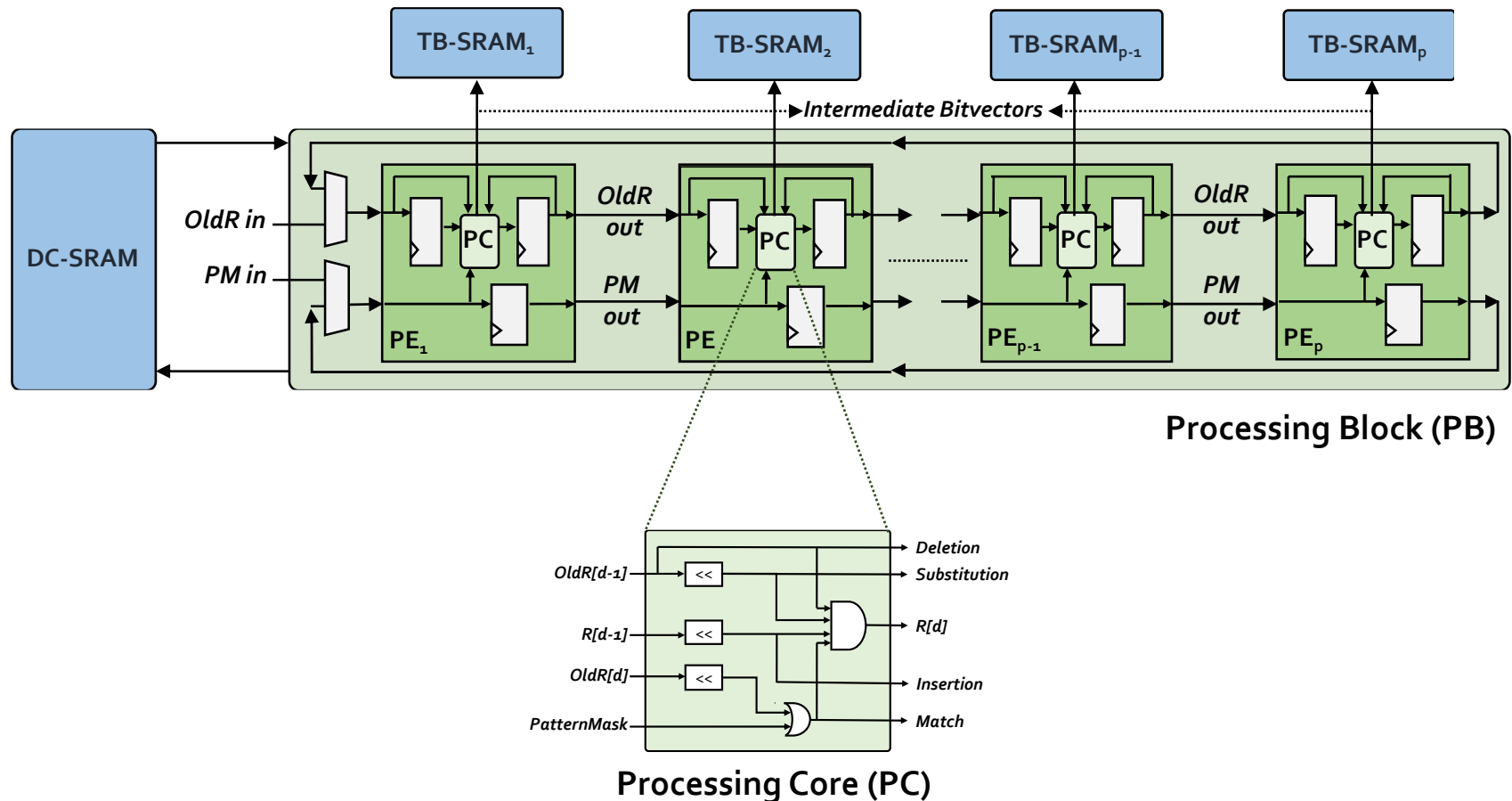
→ Match **the rate of computation** with **memory capacity and bandwidth**

→ **Achieve high performance and power efficiency**

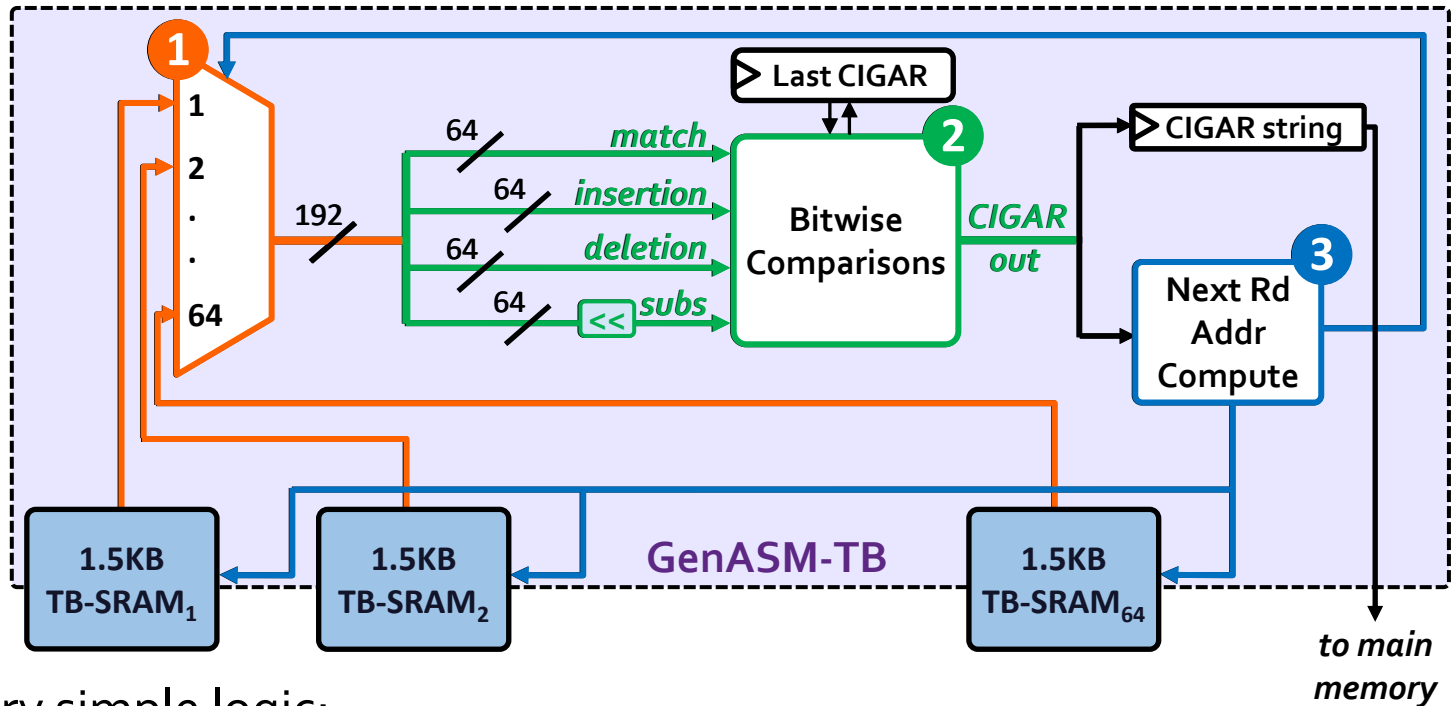
→ **Scale linearly in performance** with
the number of parallel compute units that we add to the system

GenASM-DC: Hardware Design

- ❑ Linear cyclic systolic array based accelerator
 - Designed to **maximize parallelism** and **minimize memory bandwidth and memory footprint**



GenASM-TB: Hardware Design



□ Very simple logic:

- ① Reads the bitvectors from one of the TB-SRAMs using the computed address
- ② Performs the required bitwise comparisons to find the traceback output for the current position
- ③ Computes the next TB-SRAM address to read the new set of bitvectors

Use Cases of GenASM

(1) Read Alignment Step of Read Mapping

- Find the **optimal alignment** of how reads map to candidate reference regions

(2) Pre-Alignment Filtering for Short Reads

- Quickly identify and **filter out the unlikely** candidate reference regions for each read

(3) Edit Distance Calculation

- Measure the **similarity** or **distance** between two sequences

- We also discuss **other possible use cases of GenASM** in our paper:
 - Read-to-read overlap finding, hash-table based indexing, whole genome alignment, generic text search

Outline

- ❑ Introduction
- ❑ Background
 - Genome Sequencing & Genome Sequence Analysis
 - Approximate String Matching (ASM)
 - ASM with Bitap Algorithm
- ❑ GenASM: ASM Acceleration Framework
 - GenASM Algorithm
 - GenASM Hardware Design
 - Use Cases of GenASM
- ❑ **Evaluation**
- ❑ Conclusion

Evaluation Methodology

- ❑ We evaluate GenASM using:
 - Synthesized SystemVerilog models of the GenASM-DC and GenASM-TB accelerator datapaths
 - Detailed simulation-based performance modeling

- ❑ 16GB HMC-like 3D-stacked DRAM architecture
 - 32 vaults
 - 256GB/s of internal bandwidth, clock frequency of 1.25GHz
 - In order to achieve high parallelism and low power-consumption
 - Within each vault, the logic layer contains a GenASM-DC accelerator, its associated DC-SRAM, a GenASM-TB accelerator, and TB-SRAMs.

Evaluation Methodology (cont'd.)

	SW Baselines	HW Baselines
Read Alignment	Minimap2 ¹ BWA-MEM ²	GACT (Darwin) ³ SillaX (GenAx) ⁴
Pre-Alignment Filtering	—	Shouji ⁵
Edit Distance Calculation	Edlib ⁶	ASAP ⁷

[1] H. Li. "Minimap2: Pairwise Alignment for Nucleotide Sequences." In *Bioinformatics*, 2018.

[2] H. Li. "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM." In *arXiv*, 2013.

[3] Y. Turakhia et al. "Darwin: A genomics co-processor provides up to 15,000 x acceleration on long read assembly." In *ASPLOS*, 2018.

[4] D. Fujiki et al. "GenAx: A genome sequencing accelerator." In *ISCA*, 2018.

[5] M. Alser. "Shouji: A fast and efficient pre-alignment filter for sequence alignment." In *Bioinformatics*, 2019.

[6] M. Šošić et al. "Edlib: A C/C++ library for fast, exact sequence alignment using edit distance." In *Bioinformatics*, 2017.

[7] S.S. Banerjee et al. "ASAP: Accelerated short-read alignment on programmable hardware." In *TC*, 2018.

Evaluation Methodology (cont'd.)

□ **For Use Case 1: Read Alignment**, we compare GenASM with:

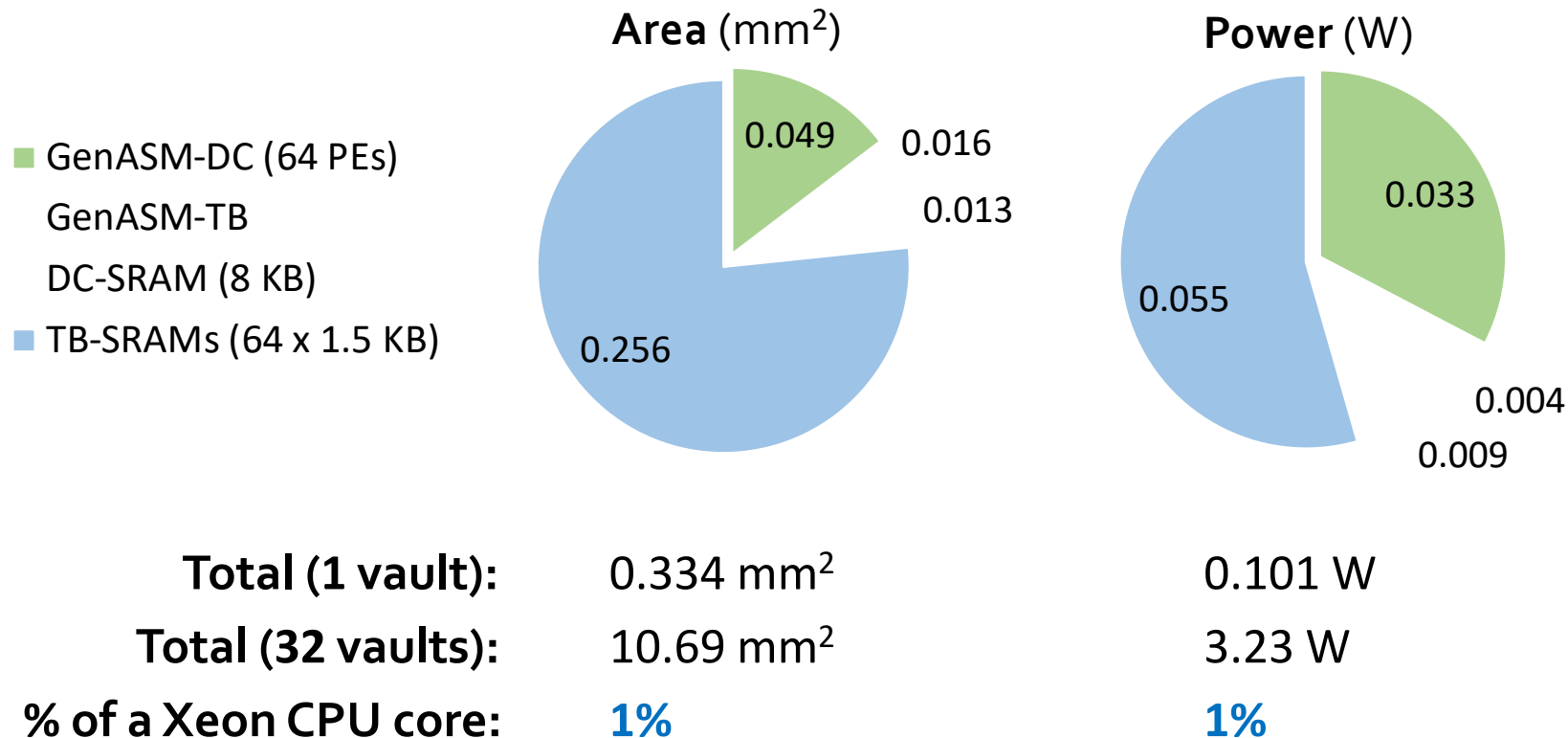
- **Minimap2** and **BWA-MEM** (state-of-the-art **SW**)
 - Running on Intel® Xeon® Gold 6126 CPU (12-core) operating @2.60GHz with 64GB DDR4 memory
 - Using two simulated datasets:
 - Long ONT and PacBio reads: **10Kbp reads, 10-15% error rate**
 - Short Illumina reads: **100-250bp reads, 5% error rate**
- **GACT of Darwin** and **SillaX of GenAx** (state-of-the-art **HW**)
 - Open-source RTL for GACT
 - Data reported by the original work for SillaX
 - GACT is best for **long reads**, SillaX is best for **short reads**

Evaluation Methodology (cont'd.)

- ❑ **For Use Case 2: Pre-Alignment Filtering**, we compare GenASM with:
 - **Shouji** (state-of-the-art **HW** – FPGA-based filter)
 - Using two datasets provided as test cases:
 - 100bp reference-read pairs with an edit distance threshold of 5
 - 250bp reference-read pairs with an edit distance threshold of 15
- ❑ **For Use Case 3: Edit Distance Calculation**, we compare GenASM with:
 - **Edlib** (state-of-the-art **SW**)
 - Using two 100Kbp and 1Mbp sequences with similarity ranging between 60%-99%
 - **ASAP** (state-of-the-art **HW** – FPGA-based accelerator)
 - Using data reported by the original work

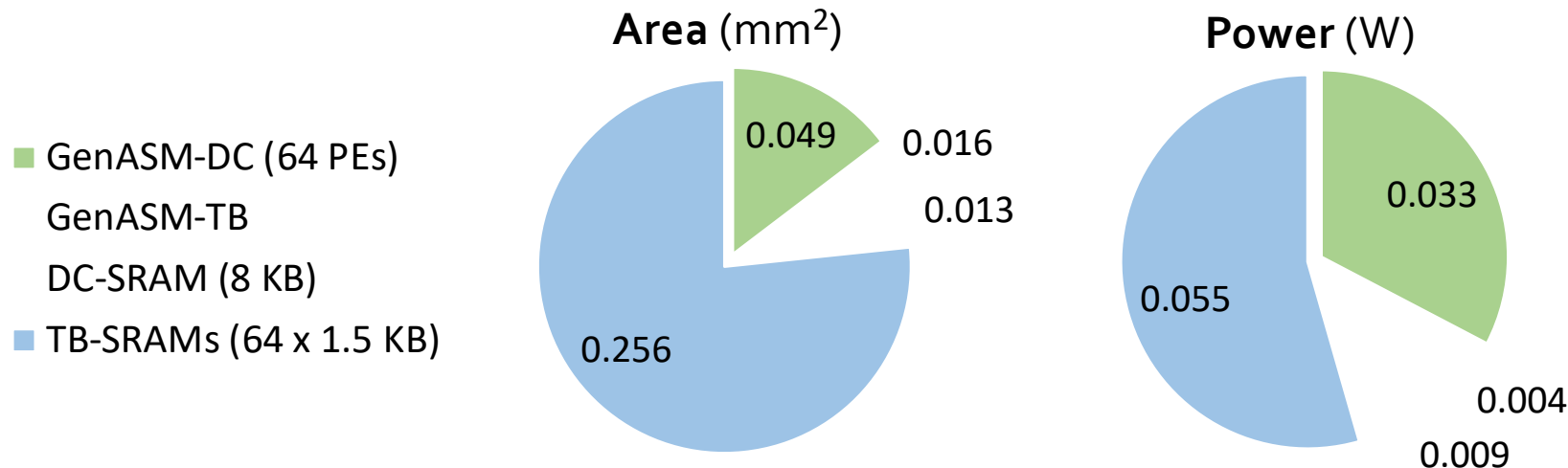
Key Results – Area and Power

- Based on our **synthesis** of **GenASM-DC** and **GenASM-TB** accelerator datapaths using the Synopsys Design Compiler with a **28nm** LP process:
 - Both GenASM-DC and GenASM-TB operate **@ 1GHz**



Key Results – Area and Power

- Based on our **synthesis** of **GenASM-DC** and **GenASM-TB** accelerator datapaths using the Synopsys Design Compiler with a **28nm** LP process:
 - Both GenASM-DC and GenASM-TB operate **@ 1GHz**



GenASM has low area and power overheads

Key Results – Use Case 1

(1) Read Alignment Step of Read Mapping

- Find the **optimal alignment** of how reads map to candidate reference regions

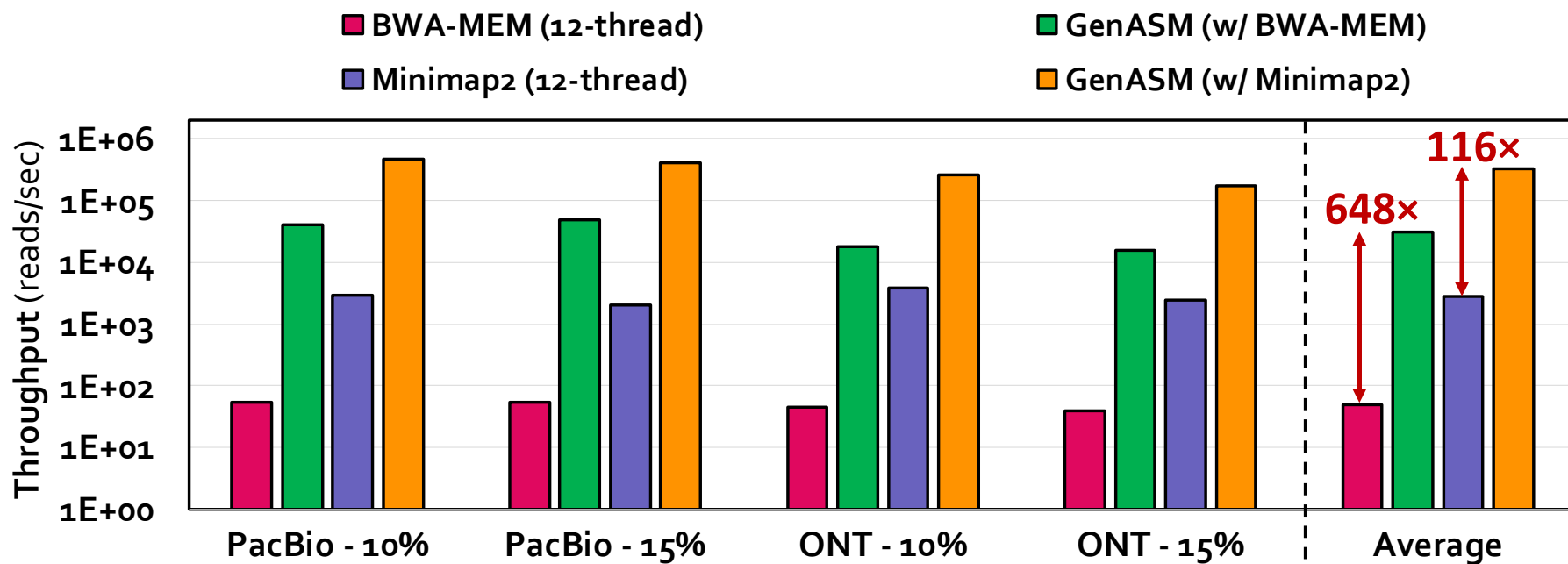
(2) Pre-Alignment Filtering for Short Reads

- Quickly identify and filter out the unlikely candidate reference regions for each read

(3) Edit Distance Calculation

- Measure the similarity or distance between two sequences

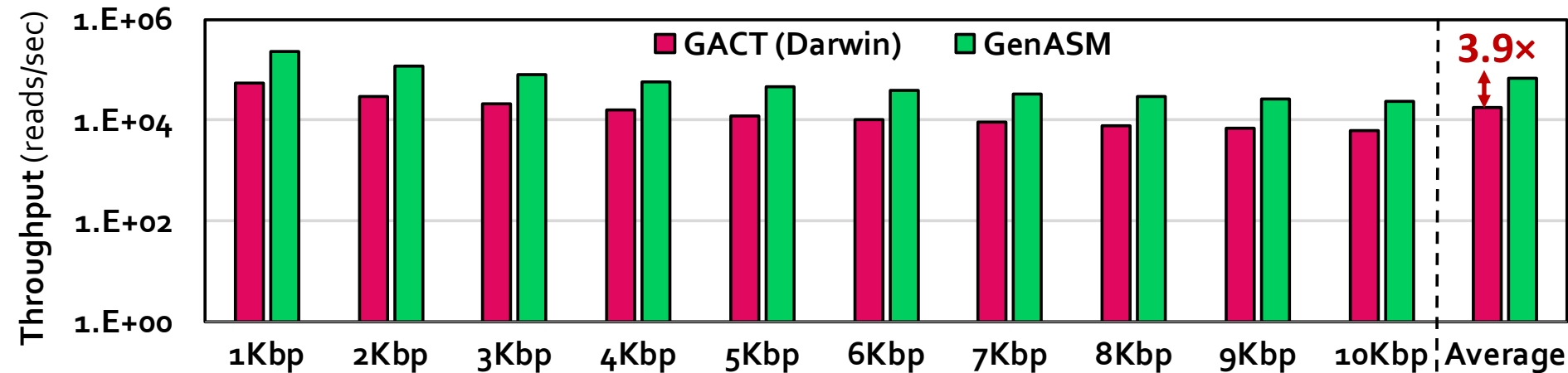
Key Results – Use Case 1 (Long Reads)



SW

GenASM achieves **648x** and **116x** speedup over 12-thread runs of BWA-MEM and Minimap2, while **reducing power consumption by 34x and 37x**

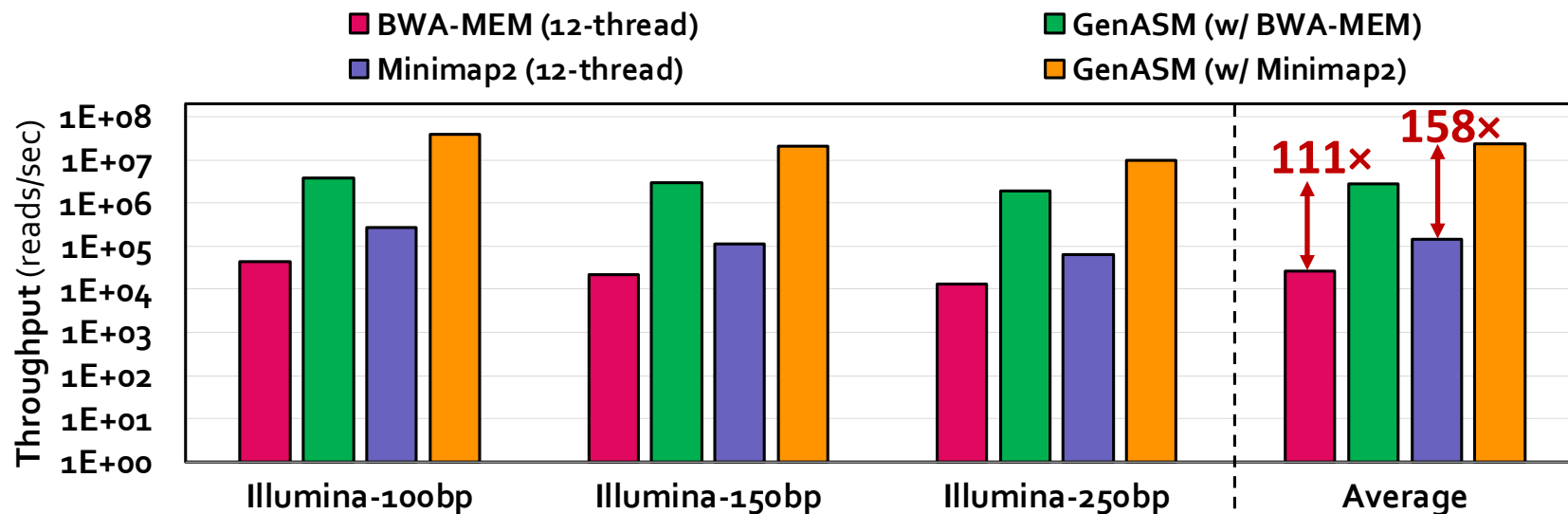
Key Results – Use Case 1 (Long Reads)



HW

GenASM provides **3.9× better throughput**,
6.6× the throughput per unit area, and
10.5× the throughput per unit power,
compared to GACT of Darwin

Key Results – Use Case 1 (Short Reads)



SW GenASM achieves **111x and 158x speedup** over 12-thread runs of BWA-MEM and Minimap2, while **reducing power consumption by 33x and 31x**

HW GenASM provides **1.9x better throughput** and uses **63% less logic area** and **82% less logic power**, compared to SillaX of GenAx

Key Results – Use Case 2

(1) Read Alignment Step of Read Mapping

- Find the optimal alignment of how reads map to candidate reference regions

(2) Pre-Alignment Filtering for Short Reads

- Quickly identify and filter out the unlikely candidate reference regions for each read

(3) Edit Distance Calculation

- Measure the similarity or distance between two sequences

Key Results – Use Case 2

❑ Compared to Shouji:

- **3.7×** speedup
- **1.7×** less power consumption
- **False accept rate of 0.02%** for GenASM vs. 4% for Shouji
- **False reject rate of 0%** for both GenASM and Shouji

HW

GenASM is **more efficient in terms of both speed and power consumption,** while **significantly improving the accuracy** of pre-alignment filtering

Key Results – Use Case 3

(1) Read Alignment Step of Read Mapping

- Find the optimal alignment of how reads map to candidate reference regions

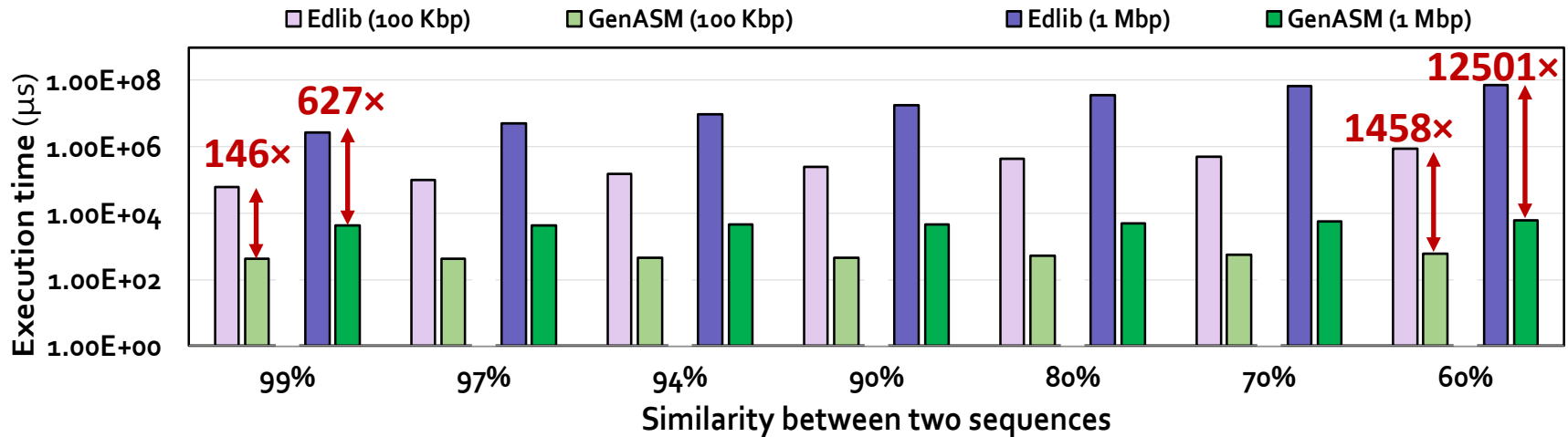
(2) Pre-Alignment Filtering for Short Reads

- Quickly identify and filter out the unlikely candidate reference regions for each read

(3) Edit Distance Calculation

- Measure the **similarity** or **distance** between two sequences

Key Results – Use Case 3



SW

GenASM provides **146 – 1458x** and **627 – 12501x speedup**, while reducing power consumption by **548x** and **582x** for 100Kbp and 1Mbp sequences, respectively, compared to Edlib

HW

GenASM provides **9.3 – 400x speedup** over ASAP, while consuming **67x less power**

Outline

❑ Introduction

❑ Background

- Genome Sequencing & Genome Sequence Analysis
- Approximate String Matching (ASM)
- ASM with Bitap Algorithm

❑ GenASM: ASM Acceleration Framework

- GenASM Algorithm
- GenASM Hardware Design
- Use Cases of GenASM

❑ Evaluation

❑ Conclusion

Additional Details in the Paper

- ❑ Details of the **GenASM-DC and GenASM-TB algorithms**
- ❑ **Big-O analysis** of the algorithms
- ❑ Detailed explanation of **evaluated use cases**
- ❑ **Evaluation methodology details**
(datasets, baselines, performance model)
- ❑ **Additional results** for the three evaluated use cases
- ❑ **Sources of improvements in GenASM**
(algorithm-level, hardware-level, technology-level)
- ❑ Discussion of **four other potential use cases** of GenASM

Conclusion

❑ Problem:

- Genome sequence analysis is bottlenecked by the **computational power** and **memory bandwidth limitations** of existing systems
- This bottleneck is particularly an issue for *approximate string matching*

❑ Key Contributions:

- **GenASM**: An approximate string matching (ASM) acceleration framework to accelerate **multiple steps of genome sequence analysis**
 - *First* to enhance and accelerate Bitap for ASM with genomic sequences
 - *Co-design* of our modified **scalable** and **memory-efficient** algorithms with **low-power** and **area-efficient** hardware accelerators
 - Evaluation of three different use cases: **read alignment**, **pre-alignment filtering**, **edit distance calculation**

- ❑ **Key Results**: GenASM is **significantly more efficient** for all the three use cases (in terms of **throughput** and **throughput per unit power**) than state-of-the-art **software** and **hardware** baselines

GenASM [MICRO 2020]

Damla Senol Cali, Gurpreet S. Kalsi, Zülal Bingöl, Can Firtina, Lavanya Subramanian, Jeremie S. Kim, Rachata Ausavarungnirun, Mohammed Alser, Juan Gomez-Luna, Amirali Boroumand, Anant Nori, Allison Scibisz, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu,

"GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis"

Proceedings of the [53rd International Symposium on Microarchitecture \(MICRO\)](#), Virtual, October 2020.

GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis

Damla Senol Cali^{†✉} Gurpreet S. Kalsi[✉] Zülal Bingöl[▽] Can Firtina[◇] Lavanya Subramanian[‡] Jeremie S. Kim^{◇†}
Rachata Ausavarungnirun[○] Mohammed Alser[◇] Juan Gomez-Luna[◇] Amirali Boroumand[†] Anant Nori[✉]
Allison Scibisz[†] Sreenivas Subramoney[✉] Can Alkan[▽] Saugata Ghose^{★†} Onur Mutlu^{◇†▽}

[†]Carnegie Mellon University [✉]Processor Architecture Research Lab, Intel Labs [▽]Bilkent University [◇]ETH Zürich

[‡]Facebook [○]King Mongkut's University of Technology North Bangkok [★]University of Illinois at Urbana-Champaign

Discussion

- ❑ GenASM for **generic text search**
 - Any other use cases?
- ❑ Most efficient **porting locations** of **GenASM accelerators**
- ❑ What about **GenASM algorithms**?
 - GPU mapping?
 - FPGA mapping?
- ❑ **Portable** sequencing devices + **low-power, memory-efficient** designs for sequence analysis
- ❑ **HW/SW co-design** for other emerging applications/domains

P&S Mobile Genomics

Lecture 12: GenASM

Dr. Damla Senol Cali

ETH Zurich

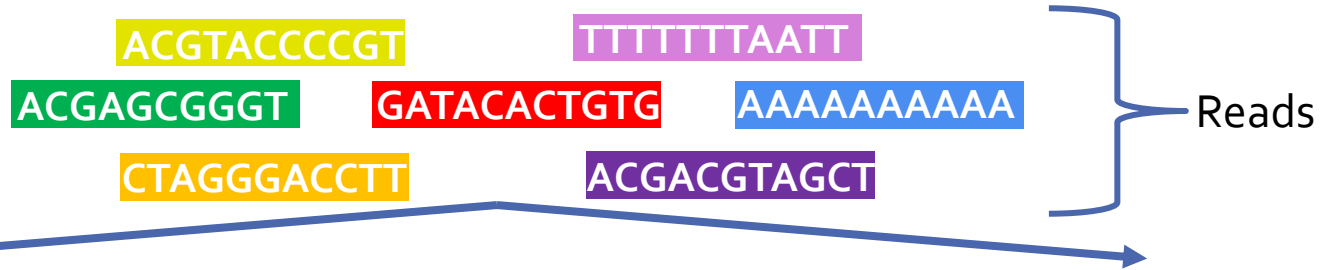
Fall 2022

16 January 2023

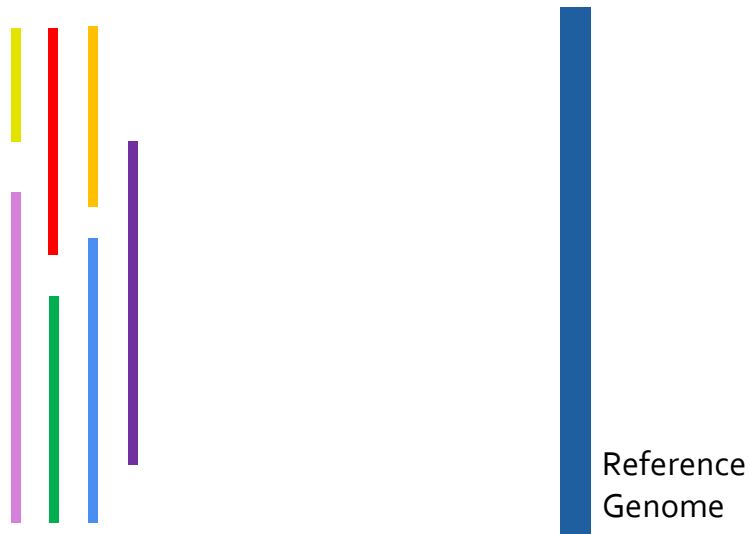
Backup Slides

(GenASM)

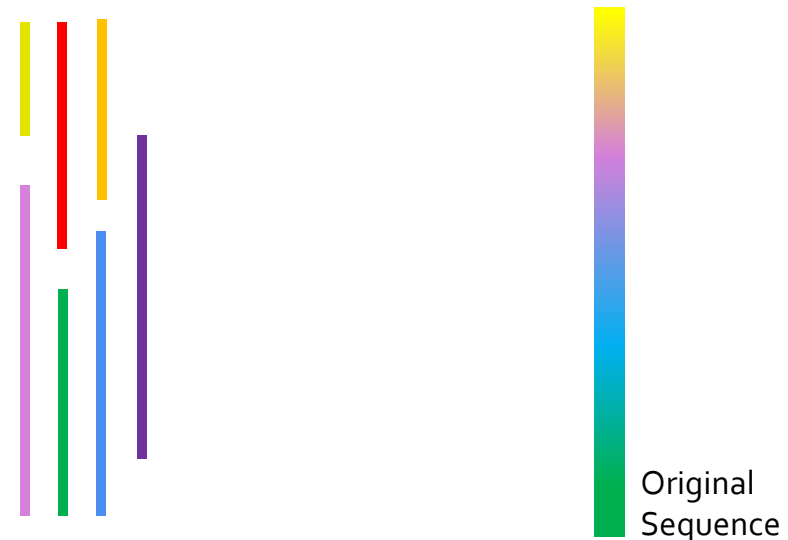
Genome Sequence Analysis



Read Mapping, method of aligning the reads against the reference genome in order to **detect matches and variations**.

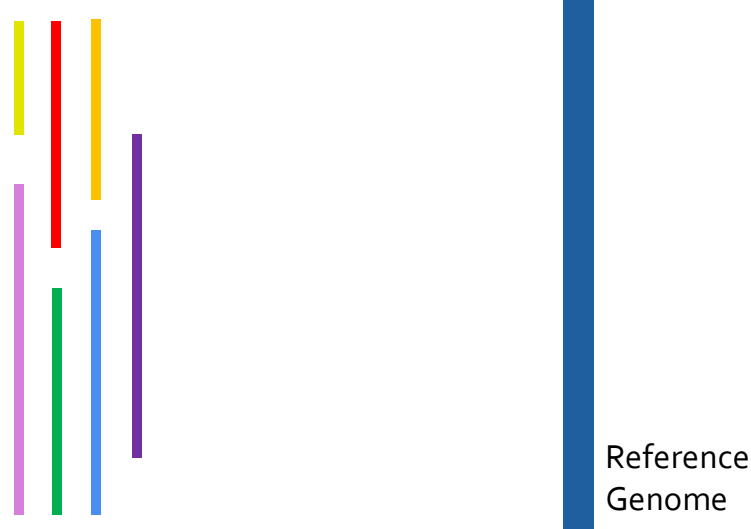


De novo Assembly, method of merging the reads in order to **construct** the original sequence.



Read Mapping

- ❑ **Read mapping:** *First key step* in genome sequence analysis
 - Align reads to one or more possible locations within the reference genome and
 - Find the matches and differences between the read and the reference genome segment at that location



Approximate String Matching (ASM)

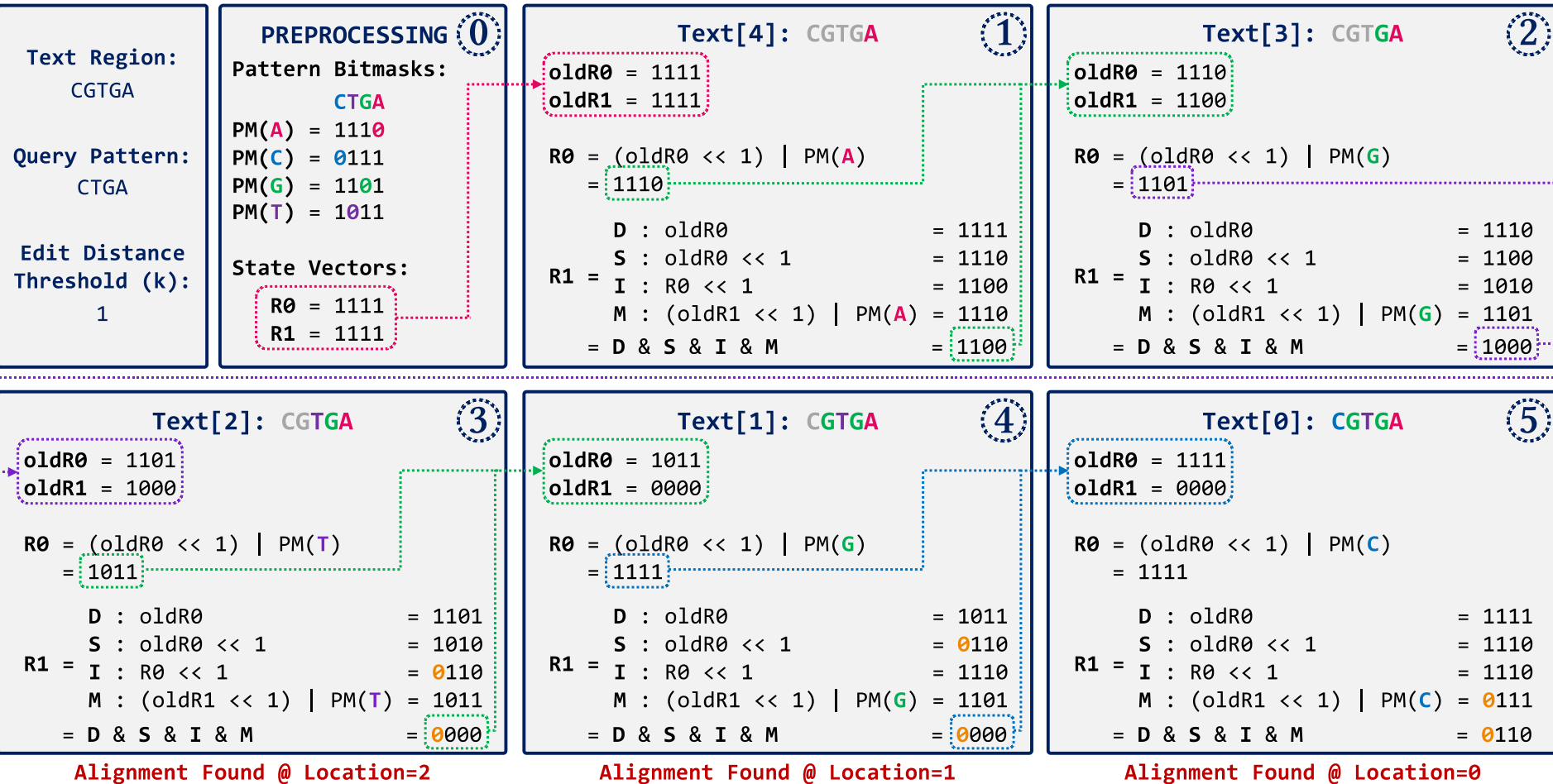
Approximate string matching algorithms:

- ❑ *Smith-Waterman (SW)* algorithm [Smith+, Advances in Applied Mathematics 1981]
 - Dynamic programming (DP) algorithm, with quadratic time and space complexity
 - Common algorithm used by read mappers
- ❑ *Myers' bitvector* algorithm [Myers, Journal of the ACM 1999]
 - Transformed version of SW algorithm into bitvectors and bitwise operations
- ❑ *Bitap* algorithm [Baeza-Yates+, Communications of the ACM 1992]
 - [Wu+, Communications of the ACM 1992] extended *Bitap* to perform approximate string matching
 - Bitvectors and bitwise operations

We have focused on the *Bitap* algorithm.

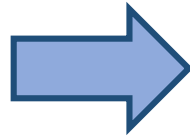
→ Reason: *Bitap* algorithm can perform ASM with **fast and simple bitwise operations**, which makes it amenable to efficient hardware acceleration.

Example for the Bitap Algorithm



Loop Unrolling in GenASM-DC

Cycle#	Thread ₁ R ₀ /1/2/..
#1	T ₀ -R ₀
...	...
#8	T ₀ -R ₇
#9	T ₁ -R ₀
...	...
#16	T ₁ -R ₇
#17	T ₂ -R ₀
...	...
#24	T ₂ -R ₇
#25	T ₃ -R ₀
...	...
#32	T ₃ -R ₇



Cycle#	Thread ₁ R ₀ /4	Thread ₂ R ₁ /5	Thread ₃ R ₂ /6	Thread ₄ R ₃ /7
#1	T ₀ -R ₀	-	-	-
#2	T ₁ -R ₀	T ₀ -R ₁	-	-
#3	T ₂ -R ₀	T ₁ -R ₁	T ₀ -R ₂	-
#4	T ₃ -R ₀	T ₂ -R ₁	T ₁ -R ₂	T ₀ -R ₃
#5	T ₀ -R ₄	T ₃ -R ₁	T ₂ -R ₂	T ₁ -R ₃
#6	T ₁ -R ₄	T ₀ -R ₅	T ₃ -R ₂	T ₂ -R ₃
#7	T ₂ -R ₄	T ₁ -R ₅	T ₀ -R ₆	T ₃ -R ₃
#8	T ₃ -R ₄	T ₂ -R ₅	T ₁ -R ₆	T ₀ -R ₇
#9	-	T ₃ -R ₅	T ₂ -R ₆	T ₁ -R ₇
#10	-	-	T ₃ -R ₆	T ₂ -R ₇
#11	-	-	-	T ₃ -R ₇

 data *written to memory*
 data *read from memory*

target cell (R_d)
 cells target cell depends on ($oldR_d, R_{d-1}, oldR_{d-1}$)

Traceback Example with GenASM-TB

Deletion Example (Text Location=0)

(a)

Text[0]: C	Text[1]: G	Text[2]: T	Text[3]: G	Text[4]: A
$\begin{pmatrix} R0- & : & \dots \\ R1-M & : & 0111 \end{pmatrix}$	$\begin{pmatrix} R0- & : & \dots \\ R1-D & : & 1011 \end{pmatrix}$	$\begin{pmatrix} R0-M & : & 1011 \\ R1- & : & \dots \end{pmatrix}$	$\begin{pmatrix} R0-M & : & 1101 \\ R1- & : & \dots \end{pmatrix}$	$\begin{pmatrix} R0-M & : & 1110 \\ R1- & : & \dots \end{pmatrix}$
Match(C)	Del(-)	Match(T)	Match(G)	Match(A)
<3,0,1>	<2,1,1>	<2,2,0>	<1,3,0>	<0,4,0>

Substitution Example (Text Location=1)

(b)

Text[1]: G	Text[2]: T	Text[3]: G	Text[4]: A
$\begin{pmatrix} R0- & : & \dots \\ R1-S & : & 0110 \end{pmatrix}$	$\begin{pmatrix} R0-M & : & 1011 \\ R1- & : & \dots \end{pmatrix}$	$\begin{pmatrix} R0-M & : & 1101 \\ R1- & : & \dots \end{pmatrix}$	$\begin{pmatrix} R0-M & : & 1110 \\ R1- & : & \dots \end{pmatrix}$
Subs(C)	Match(T)	Match(G)	Match(A)
<3,1,1>	<2,2,0>	<1,3,0>	<0,4,0>

Insertion Example (Text Location=2)

(c)

Text[-]	Text[2]: T	Text[3]: G	Text[4]: A
$\begin{pmatrix} R0- & : & \dots \\ R1-I & : & 0110 \end{pmatrix}$	$\begin{pmatrix} R0-M & : & 1011 \\ R1- & : & \dots \end{pmatrix}$	$\begin{pmatrix} R0-M & : & 1101 \\ R1- & : & \dots \end{pmatrix}$	$\begin{pmatrix} R0-M & : & 1110 \\ R1- & : & \dots \end{pmatrix}$
Ins(C)	Match(T)	Match(G)	Match(A)
<3,2,1>	<2,2,0>	<1,3,0>	<0,4,0>

Backup Slides

(Sequencing)

Short Reads vs. Long Reads

➤ Short Reads

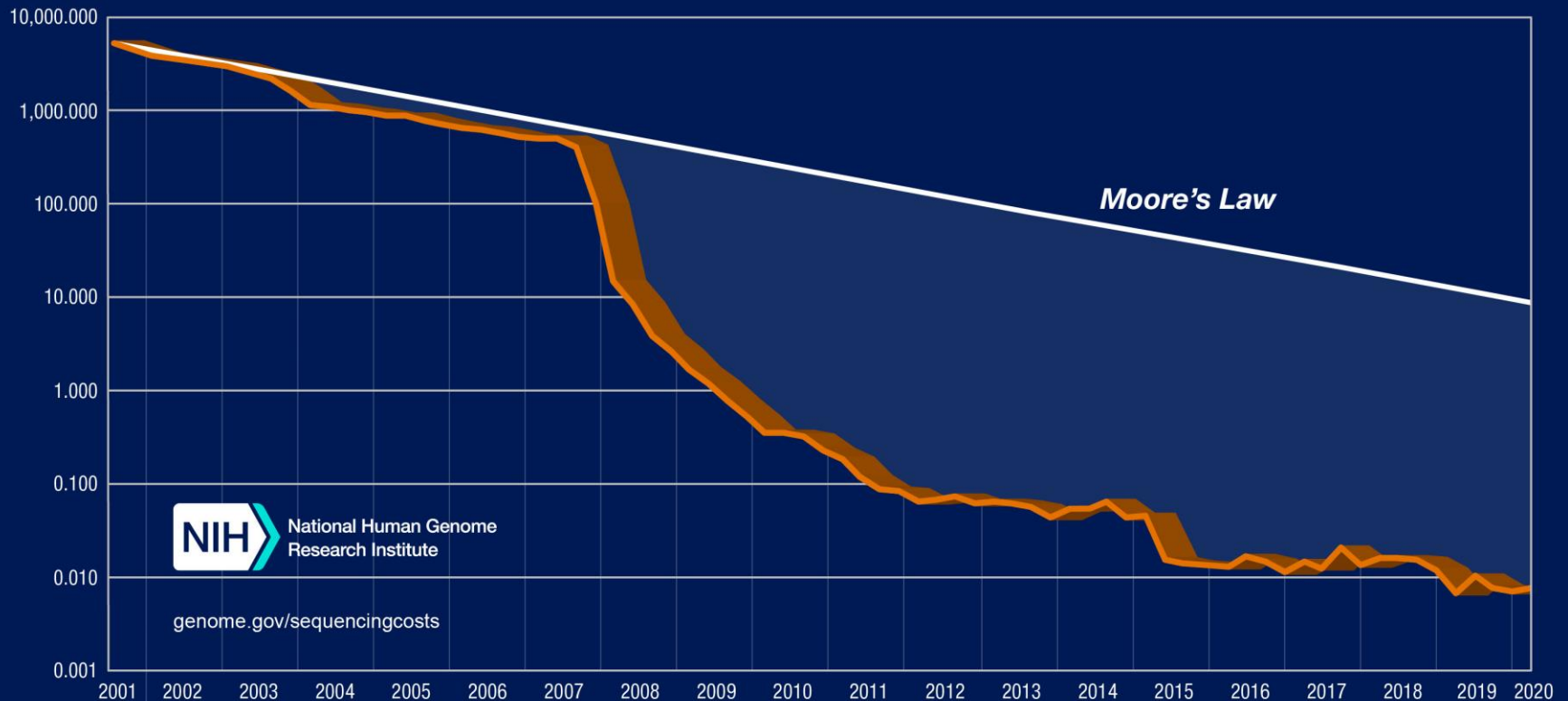
- ❑ Sequences with **tens to hundreds of bases**
- ❑ **Highly accurate** sequences
- ❑ Output of **SRS** technologies (*e.g.*, Illumina, Ion Torrent)

➤ Long reads

- ❑ Sequences with **thousands or millions of bases**
- ❑ Sequences with **high error rates**
- ❑ Output of **LRS** technologies (*e.g.*, Oxford Nanopore Technologies, PacBio)

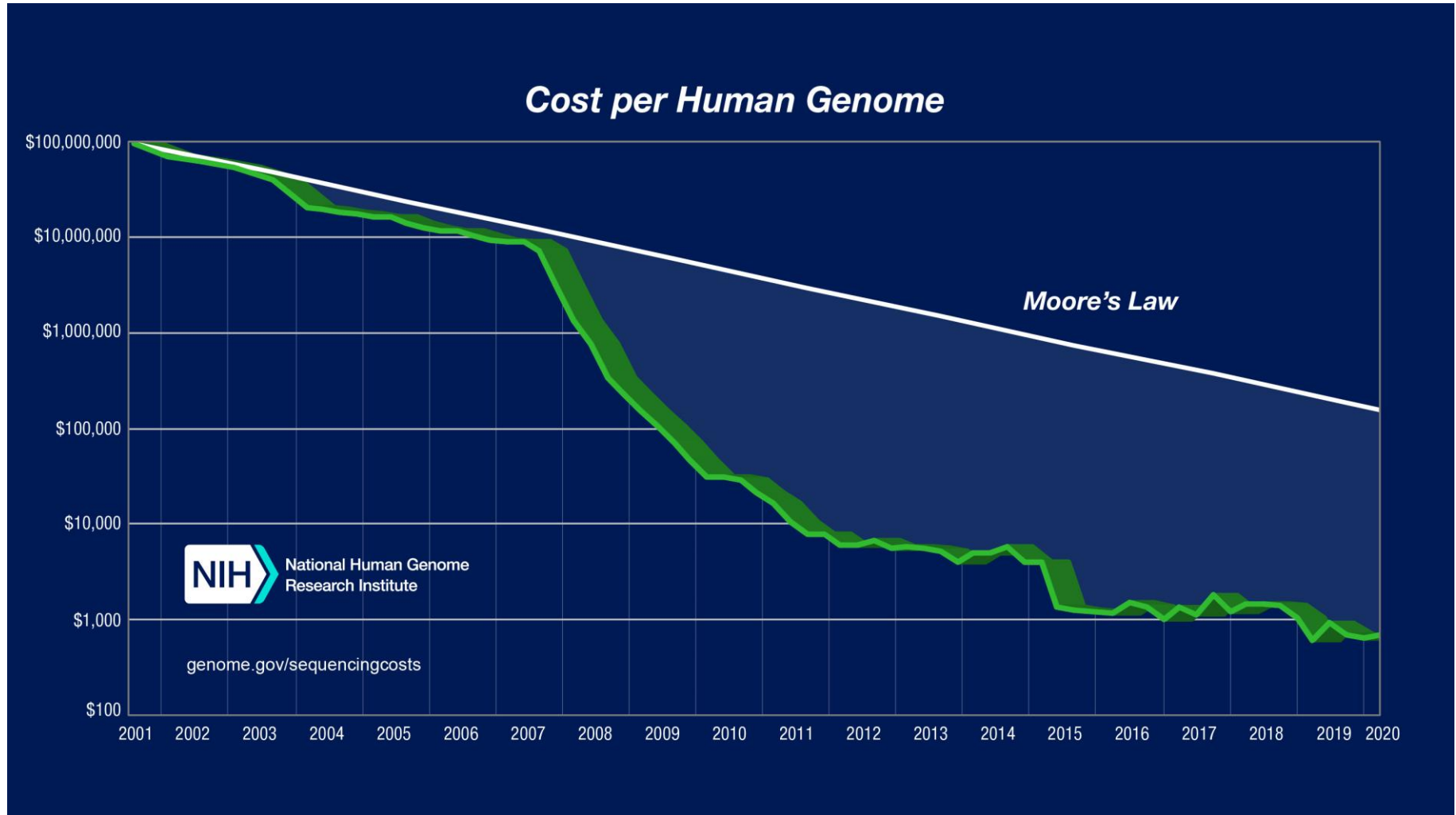
Cost of Sequencing

Cost per Raw Megabase of DNA Sequence



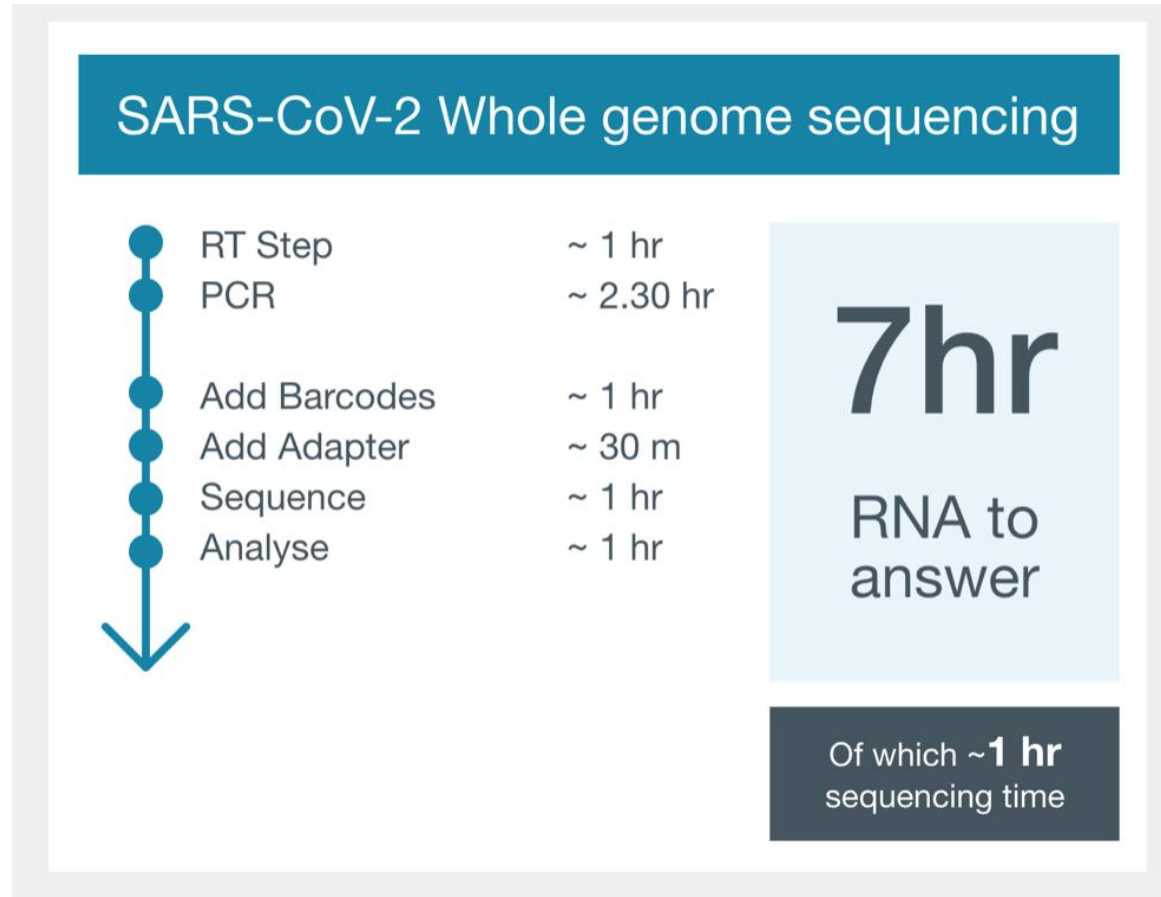
*From NIH (<https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data>)

Cost of Sequencing (cont'd.)



*From NIH (<https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data>)

COVID-19 Sequencing with ONT



- From ONT (<https://nanoporetech.com/covid-19/overview>)

COVID-19 Sequencing with ONT (cont'd.)

How are scientists using nanopore sequencing to research COVID-19?



Samples
are collected

**Validated SARS-CoV-2
RT-PCR test performed**



SARS-CoV-2 positive samples



SARS-CoV-2 negative samples:
used as negative controls

How can this be used?
Genomic epidemiology: analyse variants
& mutation rate, track spread of virus,
identify clusters of transmission

What are the results?
From RNA to full
SARS-CoV-2 consensus
sequence in ~7 hours

How?
Targeted amplification of
SARS-CoV-2 genome + multiplexed,
rapid nanopore sequencing

**Targeted SARS-CoV-2
nanopore sequencing**



**Metagenomic
nanopore sequencing**

How?
1 x RNA metagenomic
sequencing run
1 x DNA metagenomic
sequencing run

What are the results?
RNA: data for RNA viruses (including
SARS-CoV-2) + microbial transcripts
DNA: data for bacteria + DNA viruses

How can this be used?
Characterise co-infecting bacteria
& viruses, identify any correlation
of risk factors, research potential
future treatment implications

**SARS-CoV-2 Direct RNA whole
genome sequencing:** assess
viral genome in its native RNA
form and the effect of base
modifications

Immune repertoire: assess
response of the immune system to
SARS-CoV-2 infection by
sequencing of full-length immune
cell receptor genes and transcripts

**Whole human genome
sequencing:** investigate what
might cause different responses
to the virus in different people
based on their genome

What's next?



Find out more at nanoporetech.com/covid19

MinION™

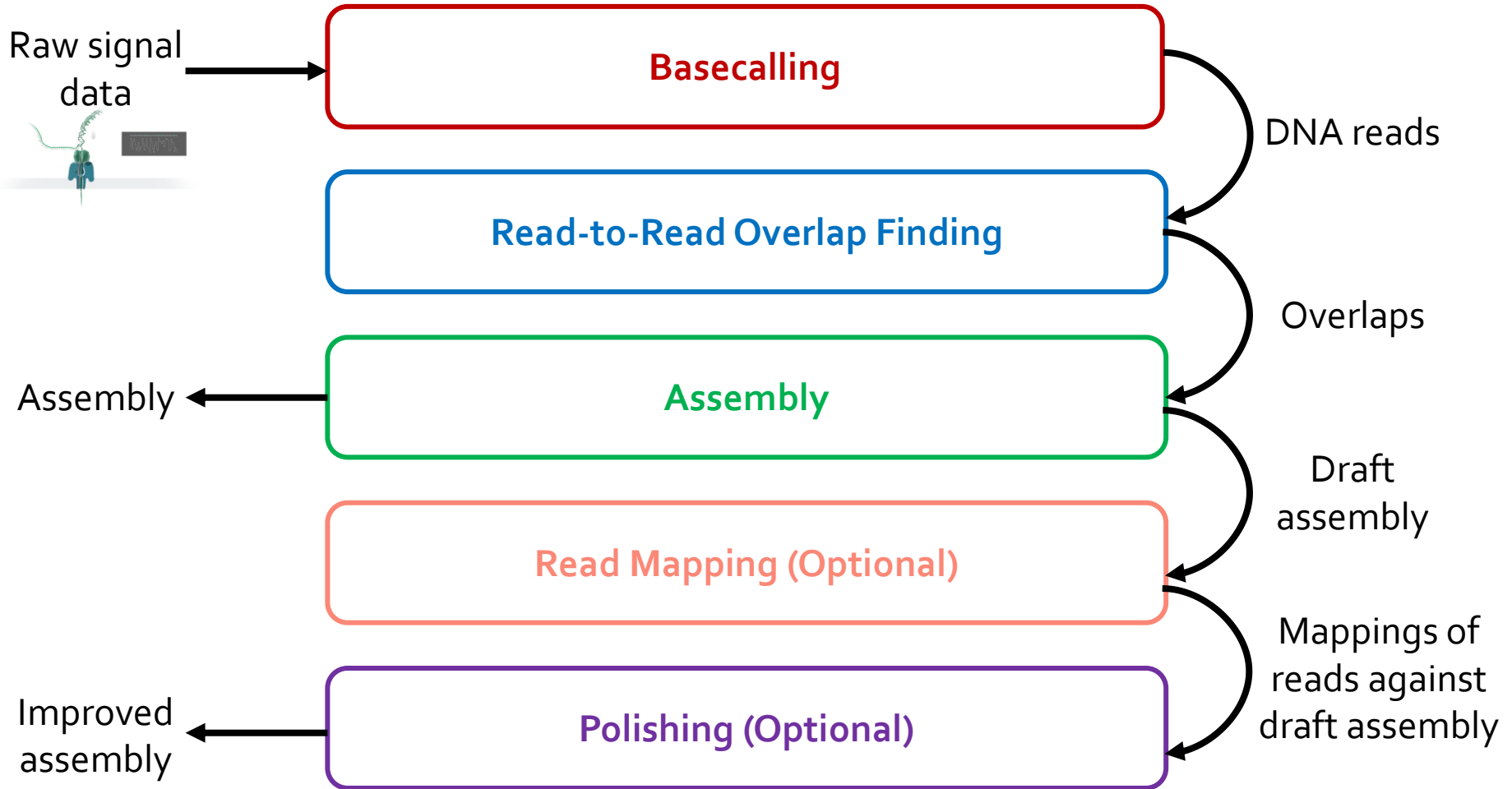
GridION™

PromethION™

Oxford Nanopore Technologies, the Wheel icon, GridION, PromethION and MinION are registered trademarks of Oxford Nanopore Technologies in various countries. © 2020 Oxford Nanopore Technologies. All rights reserved. Oxford Nanopore Technologies' products are currently for research use only. IG_1061[EN]_V1_03April2020

- From ONT (<https://nanoporetech.com/covid-19/overview>)

Nanopore Genome Assembly Pipeline



Nanopore Sequencing & Tools

Nanopore Sequencing Technology and Tools for Genome Assembly: Computational Analysis of the Current State, Bottlenecks and Future Directions

Damla Senol Cali^{1,*}, Jeremie S. Kim^{1,3}, Saugata Ghose¹, Can Alkan^{2*}
and Onur Mutlu^{3,1*}

¹Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA

²Department of Computer Engineering, Bilkent University, Bilkent, Ankara, Turkey

³Department of Computer Science, Systems Group, ETH Zürich, Zürich, Switzerland

Damla Senol Cali, Jeremie S. Kim, Saugata Ghose, Can Alkan, and Onur Mutlu. **"Nanopore Sequencing Technology and Tools for Genome Assembly: Computational Analysis of the Current State, Bottlenecks and Future Directions."** *Briefings in Bioinformatics* (2018).



BiB Version



arXiv Version